Generar un arbre a partir dels seus recorregurs en inordre i postordre X27658_ca

Preliminars

Recordeu que el recorregut en **inordre** d'un arbre és la llista dels nodes de l'arbre ordenada com segueix: en primer lloc, el recorregut en inordre del fill esquerra de l'arbre, després l'arrel de l'arbre, i després el recorregut en inordre del fill dret de l'arbre. En altres paraules:

$$inordre(x(t_1, t_2)) = inordre(t_1) \cdot x \cdot inordre(t_2)$$
$$inordre(()) = ()$$

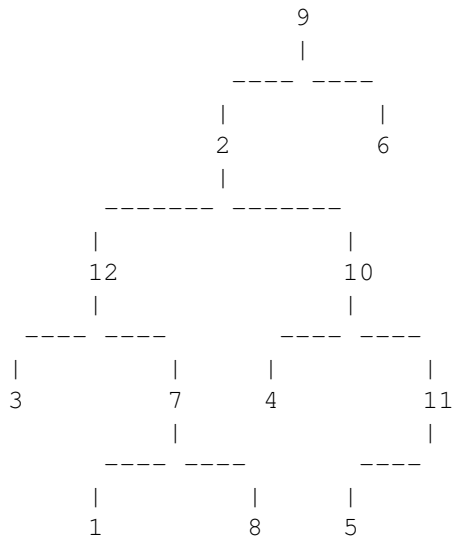
és a dir, l'*inordre* de l'arbre buit és l'arbre buit.

Recordeu també que el recorregut en **postordre** d'un arbre és la llista dels nodes de l'arbre ordenada com segueix: en primer lloc, el recorregut en postordre del fill esquerra de l'arbre, després el recorregut en postordre del fill dret de l'arbre, i finalment l'arrel de l'arbre. En altres paraules:

$$postordre(x(t_1, t_2)) = postordre(t_1) \cdot postordre(t_2) \cdot x$$
$$postordre(()) = ()$$

és a dir, el postordre de l'arbre buit és l'arbre buit.

Per exemple, considereu el següent arbre, que té 12 nodes amb valors $1, 2, \ldots, 12$:



El seu recorregut en inordre és: 3, 12, 1, 7, 8, 2, 4, 10, 5, 11, 9, 6.
El seu recorregut en postordre és: 3, 1, 8, 7, 12, 4, 5, 11, 10, 2, 6, 9.

Ara bé, seria possible, si només coneguéssim els recorreguts en inordre i postordre, reconstruir l'arbre original?

Exercici:

Heu d'implementar un programa que llegeix varis casos d'entrada. Cada cas comença amb la mida n d'un arbre binari d'enters desconegut, i ve seguit dels recorreguts en inordre i

postordre d'aquest arbre desconegut, a on cadascun d'aquests recorreguts consisteix en una llista amb tots els valors possibles entre 1 i n. Amb aquesta informació, haureu de construir l'arbre i escriure'l per la sortida.

Fixeu-vos que l'enunciat d'aquest exercici ja ofereix el fitxer `BinTree.hh`, que haureu d'utilitzar per a compilar. Només cal que creeu `main.cc`, posant-hi els includes que calguin i implementant el programa i les funcions que calguin per tal de solucionar l'exercici. Només cal que pugeu `main.cc` al jutge.

Entrada

La primera linia de l'entrada descriu el format en el que es descriuen els arbres de sortida, o bé INLINEFORMAT o bé VISUALFORMAT. Després ve una línia en blanc. Després venen un nombre arbitrari de casos. Cada cas consisteix en una primera línia amb un nombre n, una segona línia amb els nodes 1 i n en inordre i separats per espais en blanc, i una tercera línia amb els nodes 1 i n en postordre i separats per espais en blanc. Després de cada cas hi ha una línia en blanc.

Sortida

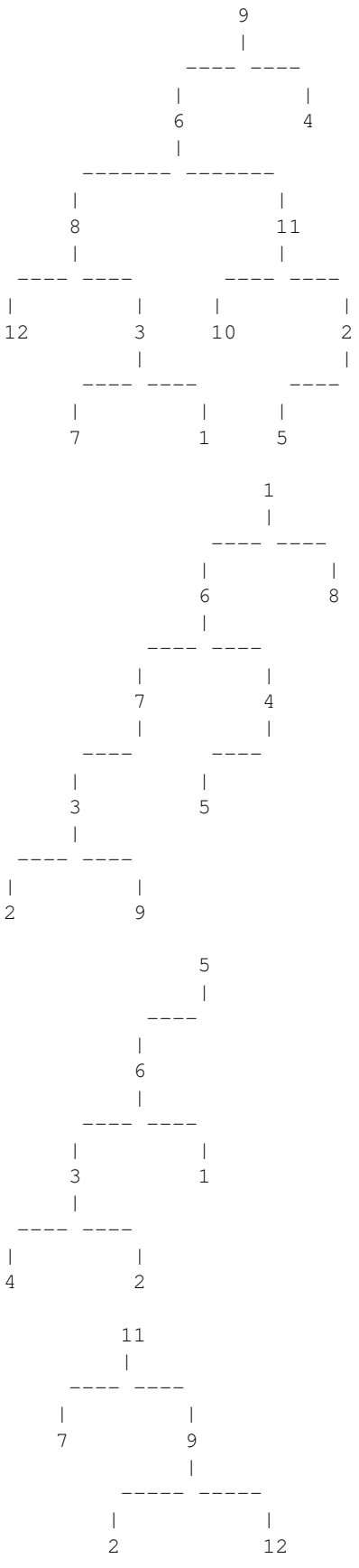
Per a cada cas, cal escriure l'arbre binari amb n nodes que contenen tots els valors entre 1 i n, i tal que el seu recorregut en inordre és la primera llista de valors, i el seu recorregut en postordre és la segona llista de valors.

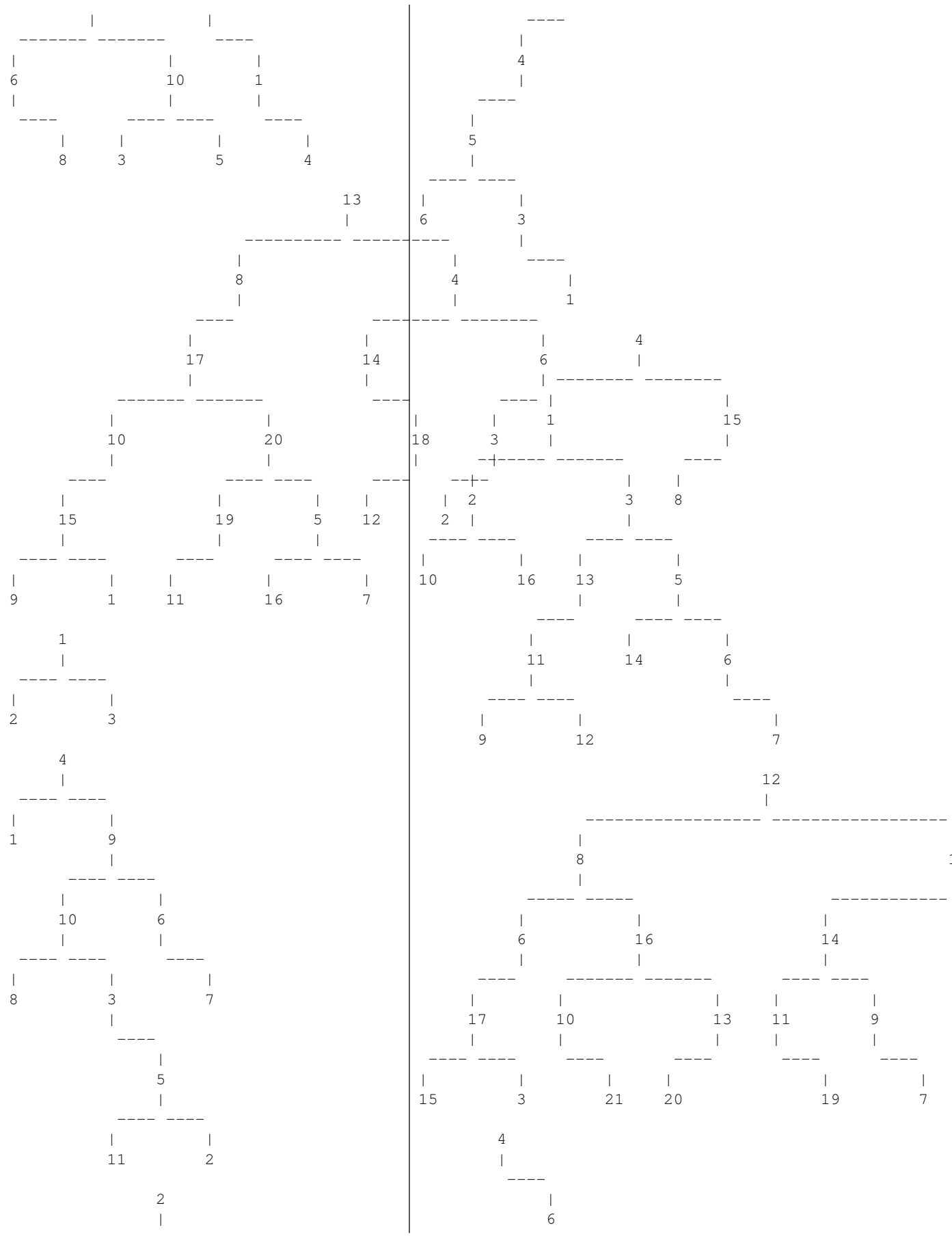
Exemple d'entrada 1

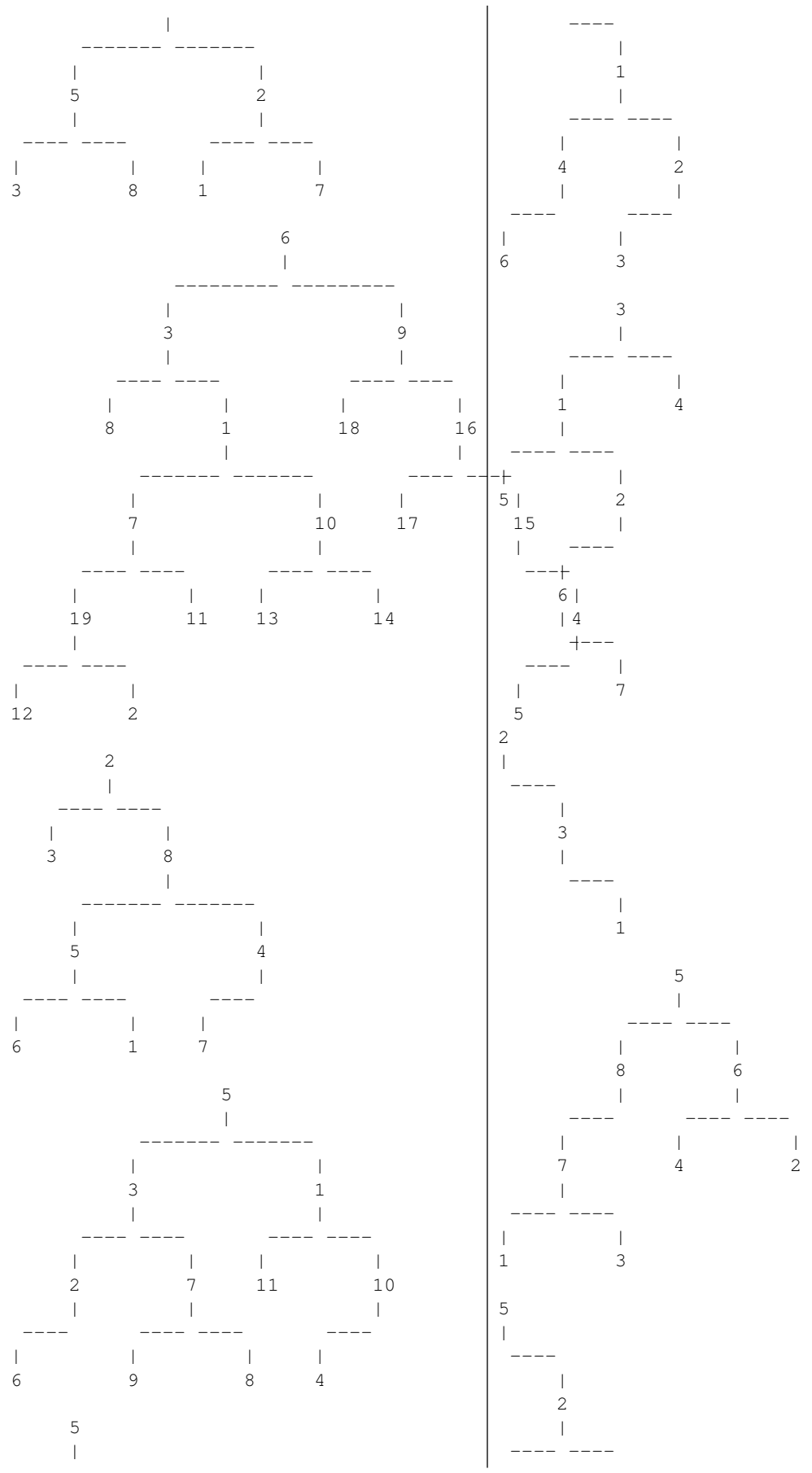
```
VISUALFORMAT

12
12 8 7 3 1 6 10 11 5 2 9 4
12 7 1 3 8 10 5 2 11 6 4 9

9
2 3 9 7 6 5 4 1 8
2 9 3 7 5 4 6 8 1

6
4 3 2 6 1 5
4 2 3 1 6 5

12
7 11 6 8 2 3 10 5 9 12 1 4
7 8 6 3 5 10 2 4 1 12 9 11

20
9 15 1 10 17 11 19 20 16 5 7 8 13 14 12 18 4 2 3 6
9 1 15 10 11 19 16 7 5 20 17 8 12 18 14 2 3 6 4 13

3
2 1 3
2 3 1

11
1 4 8 10 3 11 5 2 9 6 7
1 8 11 2 5 3 10 7 6 9 4

6
6 5 3 1 4 2
6 1 3 5 4 2

16
10 2 16 1 9 11 12 13 3 14 5 6 7 4 8 15
10 16 2 9 12 11 13 14 7 6 5 3 1 8 15 4

21
15 17 3 6 8 10 21 16 20 13 12 11 19 14 9 7 1 5 2 18 4
15 3 17 6 21 10 20 13 16 8 19 11 7 9 14 5 18 2 4 1 12

8
4 3 5 8 6 1 2 7
3 8 5 1 7 2 6 4

19
8 3 12 19 2 7 11 1 13 10 14 6 18 9 17 16 15 5 4
8 12 2 19 11 7 13 14 10 1 3 18 17 5 4 15 16 9 6

8
3 2 6 5 1 8 7 4
3 6 1 5 7 4 8 2

11
6 2 3 9 7 8 5 11 1 4 10
6 2 9 8 7 3 11 4 10 1 5

6
5 6 4 1 3 2
6 4 3 2 1 5

7
```

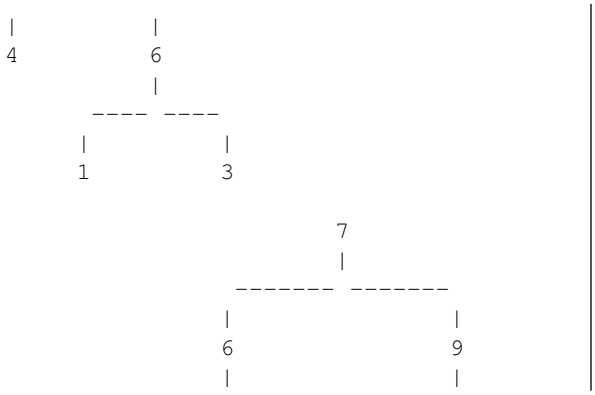
```
5 1 6 7 2 3 4
5 7 6 2 1 4 3

3
2 3 1
1 3 2

8
1 7 3 8 5 4 6 2
1 3 7 8 4 2 6 5

6
5 4 2 1 6 3
4 1 3 6 2 5

11
10 8 5 11 6 2 7 4 1 9 3
10 8 11 5 2 6 1 4 3 9 7
```

Exemple de sortida 1









Exemple d'entrada 2

```
INLINEFORMAT

12
12 8 7 3 1 6 10 11 5 2 9 4
12 7 1 3 8 10 5 2 11 6 4 9

9
2 3 9 7 6 5 4 1 8
2 9 3 7 5 4 6 8 1

6
4 3 2 6 1 5
4 2 3 1 6 5

12
7 11 6 8 2 3 10 5 9 12 1 4
7 8 6 3 5 10 2 4 1 12 9 11

20
9 15 1 10 17 11 19 20 16 5 7 8 13 14 12 18 4 2 3 6
9 1 15 10 11 19 16 7 5 20 17 8 12 18 14 2 3 6 4 13

3
2 1 3
2 3 1

11
1 4 8 10 3 11 5 2 9 6 7
1 8 11 2 5 3 10 7 6 9 4

6
6 5 3 1 4 2
6 1 3 5 4 2

16
10 2 16 1 9 11 12 13 3 14 5 6 7 4 8 15
10 16 2 9 12 11 13 14 7 6 5 3 1 8 15 4

21
15 17 3 6 8 10 21 16 20 13 12 11 19 14 9 7 1 5 2 18 4
15 3 17 6 21 10 20 13 16 8 19 11 7 9 14 5 18 2 4 1 12

8
4 3 5 8 6 1 2 7
3 8 5 1 7 2 6 4

19
8 3 12 19 2 7 11 1 13 10 14 6 18 9 17 16 15 5 4
8 12 2 19 11 7 13 14 10 1 3 18 17 5 4 15 16 9 6

8
3 2 6 5 1 8 7 4
3 6 1 5 7 4 8 2

11
6 2 3 9 7 8 5 11 1 4 10
6 2 9 8 7 3 11 4 10 1 5

6
5 6 4 1 3 2
6 4 3 2 1 5

7
5 1 6 7 2 3 4
5 7 6 2 1 4 3

3
2 3 1
1 3 2

8
1 7 3 8 5 4 6 2
1 3 7 8 4 2 6 5

6
5 4 2 1 6 3
4 1 3 6 2 5

11
10 8 5 11 6 2 7 4 1 9 3
10 8 11 5 2 6 1 4 3 9 7
```

Exemple de sortida 2

```
9(6(8(12,3(7,1)),11(10,2(5,))),4)
1(6(7(3(2,9),),4(5,)),8)
5(6(3(4,2),1),)
11(7,9(2(6(,8),10(3,5)),12(,1(,4))))
13(8(17(10(15(9,1),),20(19(11,),5(16,7))),14(,18(12,))),6(3(2,),)))
1(2,3)
4(1,9(10(8,3(,5(11,2))),6(,7)))
2(4(5(6,3(,1)),),)
4(1(2(10,16),3(13(11(9,12),),5(14,6(,7)))),15(8,))
```

```
12(8(6(17(15,3),),16(10(,21),13(20,))),1(14(11(,19),9(,
4(,6(5(3,8),2(1,7)))
6(3(8,1(7(19(12,2),11),10(13,14))),9(18,16(17,15(,4(5,)
2(3,8(5(6,1),4(7,)))
5(3(2(6,),7(9,8)),1(11,10(4,)))
5(,1(4(6,),2(3,)))
3(1(5,2(6(,7),)),4)
2(,3(,1))
5(8(7(1,3),),6(4,2))
5(,2(4,6(1,3)))
7(6(5(8(10,),11),2),9(4(,1),3))
```

Observació

AVÍS: Aquest problema requereix una anàlisi a fons abans començar a programar, perquè cal pensar detingudament l'estratègia i sobretot els detalls. La solució no és llarga però no hi arribareu per "assaig i error", és imperatiu fer servir paper i llapis primer. Els conceptes que cal tenir molt presents mentalment per poder arribar a la solució són:

- la definició recursiva de l'inordre i el postordre, i la posició de l'arrel en els dos casos,
- com fer una *immersió*, és a dir, utilitzar paràmetres a les crides recursives per establir el context en el qual s'executa cada crida.

En particular, cal notar que si tant inordre `I` com postordre `P` s'emmagatzemen en vectors, els inordres i postordres dels subarbres són subseqüències dels vectors `I` i `P`, amb posicions inicials i finals que són indexs a `I` i `P`.

La vostra solució ha de treballar amb `BinTree`, tot i que podeu utilitzar altres estructures de dades presentades al curs com a element de suport.
Per a tractar l'entrada i la sortida, podeu utilitzar aquest esquema:

```
#include <iostream>
using namespace std;
#include "BinTree.hh"
/* Pots afegir més includes */

typedef BinTree<int> BT;

/* Les funciones que vulguis */

int main() {
    string format;
    cin >> format;
    int n;
    while (cin >> n) {
        /* Llegir dues seqüències de n enters cadascuna. */

        BinTree t;
        /* Calcular t a partir de les dues seqüències. */

        t.setOutputFormat(format=="INLINEFORMAT"?  BT::INLINEFORMAT : BT::VISUA
        cout << t << endl;
```

```
    }
}
```

Avaluació sobre 10 punts:

- Solució lenta: 5 punts.
- solució ràpida: 10 punts.

Entenem com a solució ràpida una que és correcta, de cost lineal i capaç de superar els jocs de proves públics i privats. Entenem com a solució lenta una que no és ràpida, però és correcta i capaç de superar els jocs de proves públics.

Informació del problema

Autoria: PRO2

Generació: 2026-01-25T14:49:27.440Z